Based on:_Charalampidou, S. Ampatzoglou, A., Karountzos, E., and Avgeriou. P. (2019). Empirical Studies on Software Artifacts Traceability: A Mapping Study (under review).

6 Empirical Studies on Software Artifacts Traceability: A Mapping Study

Abstract

During the last decades, traceability between software artifacts has been studied in a large number of studies, from different perspectives (e.g., how to create traces? what are the benefits? etc.). This work is a secondary study on this large corpus of primary studies, focusing on empirical studies on software artifact traceability, without setting any further restrictions in terms of investigating a specific domain or concrete artifacts. The study aims at exploring the goals of existing approaches as well as the empirical methods used for their evaluation. The main contributions of this mapping study are the investigation of: (a) the types of artifacts that are linked through traceability approaches and the corresponding development phases where traces are created; (b) the goals of traceability approaches and how these goals are measured; (c) the quality attributes in a software system that benefit from traceability; and (d) the research methods used (e.g. case study, experiment etc.) for validation and assessment. The results of the study suggest that requirements artifacts are dominant in traceability, that the research corpus focuses on the proposal of novel techniques for establishing traceability, whereas the main benefits are the

improvement of software correctness and extendibility. Finally, although many studies are including some empirical validation, there are still improvements to be made, and research methods that can be used more extensively. The obtained results are discussed under the prism of both researchers and practitioners and are compared against the state-of-the-art.

6.1 Introduction

The term *traceability* in the software engineering domain refers to the creation of links between software artifacts. There are two main ways of establishing traceability: (a) **trace recovery** which refers to after-the-fact traceability and is defined as the "approach to create trace links after the artifacts that they associate have been generated or manipulated" (Cleland-Huang et al, 2012); (b) **trace capture** which refers to real-time traceability and is defined as "the creation of trace links concurrently with the creation of the artifacts that they associate" (Cleland-Huang et al. 2012). Thus, they correspond to linking artifacts in a backward and forward engineering manner respectively.

There is ample evidence in support of the benefits incurred by the creation of traces among software artifacts; we discuss here three examples. Antoniol et al. (2002) investigated the use of traces between free text documentation and code either during the development or maintenance cycle. The potential benefits of such an approach include better program comprehension, easier maintenance activities, capability to assess the completeness of the product based on the traced requirements, impact analysis and a good foundation for reusing existing software. Furthermore, Alves-Foss et al. (2002) suggest that traceability among artifacts enables incremental verification and validation of correctness and dependability properties, as well as analysis of compliance with existing standards and certifications during the lifespan of the project. Sundaram et al. (2010), discuss two beneficial contexts of traceability, which are related to the time when traceability is performed. The first one concerns the process compliance and product improvement, when traceability is performed as part of an ongoing software development process. The second concerns software understanding and reuse, when traceability work is performed on completed project data, and its results do not contribute directly to the product improvement but rather are used in the product and process analysis.

During the last decades, traceability of software artifacts has emerged as a popular topic in software engineering research, with hundreds of studies investigating nu-

merous different aspects of traceability. Thus, secondary studies are required to provide an overview of the existing literature. Secondary studies offer several benefits; among others, they support practitioners and researchers in making informed decisions in selecting the most fitting approach for their needs, or in identifying gaps for further research respectively.

To address the need for secondary studies in this field, in this chapter we report on a Systematic Mapping Study (SMS) of software artifact traceability approaches. Through this study, we explore four different aspects of traceability approaches: (a) the types of artifacts that are linked through traceability approaches and the corresponding development phases where traces are created; (b) the goals of traceability approaches and how these goals are measured; (c) the quality attributes in a software system that benefit from traceability; and (d) the research methods used (e.g. case study, experiment etc.) for validation and assessment. Compared to other secondary studies on traceability, this study focuses only on empirical studies, but sets no further restrictions in terms of investigating a specific domain or concrete artifacts (more details are given in Section 6.5). Additionally it is worth mentioning that this study has selected a significantly larger set of primary studies on the topic of traceability.

The choice of conducting a systematic mapping study instead of a systematic literature review needs further justification. According to Kitchenham et al. (2011) an important criterion for selecting the most appropriate form of a secondary study is its goal and scope. The goal of a SMS is the classification and thematic analysis of the literature on a specific topic. On the other hand an SLR aims at "*identifying best practice with respect to specific procedures, technologies, methods or tools by aggregating information from comparative studies*" (Kitchenham et.al. 2011). The scope of an SLR is more focused and the papers included are usually empirical studies related to a specific research question. Additionally, the information extracted from each paper is usually about individual research outcomes. As aforementioned, this study is substantially broad as it aims at classifying primary studies in terms of linked artifacts, their goals, the benefits they provide to quality attributes, and the research methods used. Therefore a SMS is more appropriate for the goal and scope of our study. We do note that we investigate only empirical studies (which is usually the practice in SLRs instead of SMS). However, this is not uncommon; for example Budgen et al. (2008) present a set of mapping studies that report findings specifically on empirical studies rather than the research literature as a whole.

This chapter is organized as follows; In Section 6.2 we present the mapping study protocol we followed. In Section 6.3, we present the results of the mapping study and in Section 6.4 we discuss the results. In Section 6.5 we discuss related work in terms of secondary studies in the domain of software artifact traceability, and we compare it with our own study. Finally, in Section 0 we present the threats to validity and in Section 6.7 the conclusions of this work.

6.2 Study Design

This section describes the design of the proposed systematic mapping study. To design this mapping study we followed the process proposed by Petersen (Petersen et al. 2008). The essential process steps of a systematic mapping study are: (a) definition of research questions; (b) conducting the search for relevant papers; (c) screening of papers; (d) keywording of abstracts; and (e) data extraction and mapping.

6.2.1 Research Questions

The goal of the study is formulated according to the Goal-Question-Metrics (GQM) approach (Basili et al. 1994) as follows: "***Analyze*** *empirical studies on software artifact traceability approaches* ***for the purpose of*** *characterization,* ***with respect to:*** *(a) the types of artifacts that are connected through traces, (b) the expected goals of using traceability and their success indicators, (c) the quality attributes that are improved through the establishment of traces, and (d) the research methods used,* ***from the point of view of*** *software engineering practitioners and researchers*". According to the abovementioned goals, and the expected contributions (see section 6.5) we extracted four research questions (RQ), as follows:

RQ_1: *Which types of artifacts are connected and in which development phase are these artifacts created?*

With this research question we aim to investigate which types of artifacts (e.g. use cases, design artifacts, source code) are most commonly used for traceability purposes and to what artifacts they are usually linked to. In addition the phase (e.g. requirements engineering, design, implementation) in which the artifact is developed is going to be investigated. In other words, we want to understand when traces are usually created (i.e. development phase) and between which artifacts. This in turn will help in identifying potential areas of traceability that have been understudied.

RQ_2: *What are the goals of the studies and how is the achievement of these goals measured?*

To answer this research question, we classify the studies according to their main goal. We look at studies aiming at *creating* traces among software artifacts, as well as those that *maintain* traces after they have been created either manually or automatically. We include both creation and maintenance as they are considered equally important in traceability management. Finally, we present success indicators that quantify the extent to which the approaches achieve their respective goal. By answering this research question we aid practitioners in selecting the most fitting approaches for establishing and maintaining traces, and researchers in selecting pertinent measurements for validating their approaches.

RQ_3: *Which quality attributes benefit from the establishment of traces?*

Upon the establishment of traces, various benefits are expected in terms of quality attributes. In this research question we investigate the impact of traceability on quality attributes as described in ISO25010 (e.g. maintainability, correctness etc.) and provide the measures used for investigating how traces influence those qualities. The answer to this research question is expected to: (a) highlight the quality benefits that practitioners can gain by establishing and maintaining traces; and (b) identify the most interesting quality aspects that can be considered when investigating software traceability.

RQ_4: *What research methods are used for validation and assessment?*

Finally, this research question will look into the most commonly used empirical research methods to validate traceability approaches or assess the influence of traces on software quality. This may provide insight on the trends of the research methods used and can indicate ideas for future work to fill in potential gaps.

6.2.2 Search Process

Search Scope. The search procedure of a mapping study aims at identifying as many primary studies related to the research questions as possible, using an unbiased search strategy. To achieve this goal we performed an automated search process using generic digital libraries, namely: (a) IEEE Digital Library; (b) ACM Digital Library; (c) Springer Link; and (d) Science Direct. In order to select the appropriate digital libraries, we adopted the inclusion criteria by Dieste et al. (2007): content update (dynamic update with new publications); availability

(provide access to the full text of every research article); quality of results (test the accuracy of results returned from the query process using a small list of expected publications which are set by our team from empirical search); and versatility export (since there is a lot of noise in the retrieved results there is a process to filter the raw data). The selected sources are well-known and are constantly used in such type of studies in the software engineering field. We decided to exclude from the list of DL indexing mechanisms, e.g., Scopus, Google Scholar, or DBLP, to: (a) avoid extensive duplication of articles; and (b) avoid the inclusion of grey literature or non-peer-reviewed materials.

Search Terms. In a systematic mapping study, to minimize bias and to maximize the number of sources examined, a pre-defined strategy to identify potential primary studies is required. For the automated search, the search string consisted of three main parts: *trace* **AND** *empirical* **AND** *software*. For each part, a list of alternate terms was used and connected through logical **OR** to form a more expressive query. The terms related to the "trace" part had to exist in the title of the papers, since they define the domain of interest, and we expected that in the vast majority, they should be part of the title. The terms for the "empirical" and "software" parts could exist either in their title, abstract or keywords.

Regarding the "empirical" part, the alternate terms should be concrete types of empirical research methods, since often the term "empirical" is not present in the title, abstract or keywords, resulting to missing several relevant studies. Thus, we considered several names of empirical research methods found in literature, as shown in Table 6.1. The first column of the table shows the research method names we considered, while the next 6 columns indicate in which sources these research methods have been considered as empirical research.

To identify the list of sources, we initially referred to the most well-known papers and books dealing with empirical research (e.g. (Wholin et al. 2000) and (Runeson et al. 2012). However, these sources concerned specific research methods (i.e. experiments and case studies respectively). Thus we looked for papers dealing with empirical research in a more generic point of view and came up with 5 studies providing explicitly types of empirical methods. Additionally, since ESEM is the main venue where empirical community tends to publish, we collected the research methods mentioned in their call for papers too. Similarly we looked into the journal of *Empirical Software Engineering*, however no keywords, we were interested in,

were specified. Regarding the last column of the table, it shows how many times each keyword (research method) has been defined as empirical research. In our search string we used as keywords only research methods which have been considered as empirical by at least two sources (green cells).

Table 6.1: Empirical research methods found in literature

Research Methods	Easterbrook et al., 2008	de Magalhães et al., 2014	Hummel, 2014	Silva et al., 2015	ESEM	Stol et al, 2009	Count
Survey	X	X	X	X	X	X	6
Case Study	X	X	X	X	X	X	6
Action Research	X	X	X	X	X	X	6
Experiment	X	X	X		X	X	5
Ethnography	X	X	X			X	4
Field Research/Study		X			X	X	3
Grounded Theory			X			X	2
Simulation			X		X		2
Quantitative Analysis					X	X	2
Experience Report / Industrial				X	X		2
SLR			X		X		2
Theoretical/Descriptive				X			1
Meta-Analysis					X		1
Qualitative					X		1
Focus Group			X				1

We note that although the term SLR had a total sum of 2 references, we did not use the term in the search string, since we were not interested in collecting SLR studies as primary studies. Such studies would be interesting related work, but since they would not focus on the presentation or evaluation of a traceability approach they would be excluded from the list of primary studies.

Taking the above into consideration, the search string used was the following:

("trace" OR "tracing" OR "traceability")

AND

("empirical" OR "case study" OR "survey" OR "experiment" OR "action research" OR "ethnography" OR "field research" OR "field study" OR "grounded theory" OR "simulation" OR "quantitative analysis" OR "experience report" OR "industrial")

AND

("software")

The outcomes of the automated search were highly dependent on the quality of the search string used, and thus several refinements were needed before the search string could be considered as well defined. For this reason the search string was validated while being used in a subset of the venues defined in the protocol. The studies fetched from this search were analyzed aiming at verifying if they are in accordance to the objective of the mapping study and the main research questions. In cases that there were findings showing that the search string could be improved, changes were applied and the process was repeated. All the results of this research were stored using the JabRef software, an open source bibliography reference manager. The details of the papers (i.e., title, author(s), abstract, keywords, year of publication and the name of the data source) were directly exported from the digital libraries to JabRef, using a second reference management tool, i.e. Zotero.

Primary Study Validation. To ensure high relevance of the extracted primary studies with the subject of this research, we have applied the quasi gold standard process as suggested by Zhang and Babar (2010). In particular we decided to perform a manual search in the top two journals and conferences, in the domain of software engineering (TSE, TOSEM, ISCE and FSE). However, since the amount of papers in these venues was considerably high, we selected to limit the manual search starting from January 2011. Next, we compared the results of the manual

and automatic search process indicating whether the search string was capable to return relevant studies based on the research questions.

6.2.3 Study Selection (Screening)

The primary studies to be selected must be relevant to empirical investigation of software artifact traceability approaches. In line with (Dyba and Dingsoyr 2008), there are three phases of filtering the article set to produce the primary study data set. The first phase comprises of the search process (described in Section 6.2.2) returning a set of candidate primary studies. This set subsequently goes through two phases of manual inspection: in the first one the inclusion/exclusion criteria are applied on each article's title, abstract/conclusions, while on the second phase they are applied on each article's full text. The inclusion/exclusion criteria that will be used in every phase are listed below:

- Inclusion criteria:
  - The study introduces a software artifact traceability approach
  - The study evaluates a software artifact traceability approach
  - The full text of the study is available in English
- Exclusion criteria:
  - The study introduces an approach for tracing the same artifacts across versions (or)
  - The study introduces a traceability approach without stating explicitly the software artifacts that are linked (or)
  - The study is an editorial, position paper, keynote, opinion, tutorial, poster or panel (or)
  - The study is a previous version of a more complete paper about the same research

Every article selection phase was handled by the first two authors, who both checked all primary studies, and possible doubts were resolved by the third and fourth authors. For each data source mentioned in Section 6.2.2, we documented the number of papers that were returned. Also, we recorded the number of papers left for each venue after primary study selection on the basis of title and abstract. Moreover, the number of papers finally selected from each source was recorded. The number of primary studies selected in each of the three phases is shown in the table below.

Table 6.2: Overview of primary studies

Digital Library	Initial search	1st Exclusion (title, abstract, conclusions)	Final dataset
IEEE DL	431	183	**95**
ACM DL	213	94	**33**
Springer Link	31	24	**16**
Science Direct	39	19	**11**
Total	**714**	**320**	**155**

6.2.4 Keywording of Abstracts (Classification Scheme)

In (Petersen et al. 2008) the authors propose keywording of abstracts as a way to develop a classification scheme, if existing schemes do not fit, and ensure that the scheme takes into account the identified primary studies. In this study, we applied keywording in order to classify artifact tracing approaches with respect to:

- The artifacts they link.
- Their benefits and the success indicators used to measures this benefit.
- Their research setting, in terms of empirical method used and data collection methods.

Since the aforementioned information could not be obtained based only on the abstract of the study, we decided to apply the keywording technique to the articles' full texts.

6.2.5 Data Extraction & Mapping

During the data collection phase, we collected a set of variables from each primary study. Data collection was executed by the first two authors and possible conflicts were resolved by the third and fourth author. For every study, we extracted assigned values to the following variables:

[V1] Author: *the list of authors*

[V2] Year: *the year of the publication*

[V3] Title: *the title of the publication*

[V4] Source: *the library where the study was found*

[V5] Venue: *the venue where the study was published*

[V6] Type of Paper: *the type of the publication (conference / journal)*

[V7] Keywords: *the keywords reported in the publication*

[V8] Connected artifacts: *the software artifacts connected by the traceability approach*

[V9] Goal of the study: *the goal of the study in terms of research questions or expected benefits.*

[V10] Success indicators: *the means to measure the expected benefits*

[V11] Type of data analysis: *qualitative, quantitative, mixed*

[V12] Research method used: *the type of the empirical method used (case study / experiment etc.)*

Attributes [V1] – [V7] were used for Documentation reasons. All other variables were used for answering a corresponding research question. The mapping between attributes and research questions is provided in the following table, accompanied by the analysis and presentation methods used on the data.

Table 6.3: Data analysis techniques per research question

Research Question	Variables Used	Analysis & Presentation Method
RQ_1	[V8]	Count of connected artifacts (individual artifacts) Count of connected pairs of artifacts Grouping of identified artifacts
RQ_2	[V9], [V2]	Keywording (goals) → Count of different goals Trends (of goals per year) Crosstabs (goals – success indicator)

Research Question	Variables Used	Analysis & Presentation Method
RQ_3	[V9], [V10], [V11]	Keywording (affected QAs) → Count of different affected QAs Trends (of affected QAs per year) Crosstabs (QAs – success indicator)
RQ_4	[V2], [V8], [V12]	Count of research methods Trends (of research methods per year) Crosstabs (research method – pairs of artifacts RQ_1) Count (qualitative / quantitative) Crosstabs (research method – goal/benefit RQ_2)

For research question **RQ_1** we investigate which are the most frequent individual artifacts, and pairs of software artifacts connected using traceability approaches. To do so we count the sets of pairs and individual artifacts found in the literature. Additionally we group together artifacts in larger categories (e.g. different types of UML diagrams, or different forms of requirements).

To analyze the extracted goals of the primary studies in **RQ_2**, we apply the keywording technique. Keywording suggests the identification of keywords in the extracted data, in order to identify and map similar concepts (Petersen et al. 2008). Finally we report the findings based on the publication year, to show how the research topics evolved during time. The same procedure has been performed for success indicators. To do so, we do a merging process (Bafandeh Mayvan et al. 2017) to group together similar success indicators. Then we performed crosstabs matching the identified goals to the success indicators, i.e. the ways that each goal can be measured.

For **RQ_3** we reported the results in the same way as RQ_2 by replacing goals with affected quality attributes. For **RQ_4** we count the different types of empirical research methods found in the literature. Then we show the trend throughout the years to investigate what type of research is mostly conducted as the years go by, and we relate the type of research method with the pairs of artifacts usually con-

nected. Similarly, we perform crosstabs to investigate potential relations between the research methods and the goals/benefits studied. Finally we report on the type of analysis (i.e. qualitative, quantitative or mixed).

6.3 Results

In this section we present the results of this study, organized by research question. Therefore, in Section 6.3.1 we investigate the types of artifacts that are being traced along different software development phases (RQ_1). In Section 6.3.2, we present the results on RQ_2, in which we studied the goals of proposing new traceability approaches. Finally, in Section 6.3.3, we present an overview on the impact of traceability on quality attributes, whereas in Section 6.3.4 we discuss the empirical setting of the validation.

6.3.1 Types of Traceability Artifacts and Development Phases (RQ_1)

This section aims to summarize the findings of our study with respect to the artifacts that are connected with traceability links, and the development phases in which they are produced. The sub-section is organized in two parts: the first one reports artifacts in isolation, i.e. the number of studies in which the artifacts are used regardless of the artifacts they are linked to; the second presents results on the pairs of artifacts that are being connected. Such a distinction is useful to understand both the types of artifacts that are most frequently traced (e.g., if requirements are more frequently traced than tests) as well as the pairs of artifacts that are often linked. Both parts state the development phases where the artifacts are produced.

In Table 6.4, we present the top-15 most frequently investigated *individual* artifact types. For each artifact we denote the development phase in which it is produced (R: requirements engineering, D: design, I: implementation, T: testing), and the count of primary studies, in which we have identified them. We note that the level of detail in which the artifact is being presented, depends exclusively on the reporting of the primary study. For instance in the 1st row we identified the term "requirements (high or low level)" and in the 5th row the term "use case"; this is because 31 papers were explicitly referring to use cases (or provided concrete examples with use cases), whereas 70 were referring to requirements, without clarifying how they are specified (e.g., use cases, user stories, natural language, etc.). From the findings of Table 6.4 we can observe that the development phases of require-

ments engineering, design and implementation are almost equally represented in terms of count of types of artifacts (i.e. number of lines in table 4). However, design artifacts appear in fewer studies compared to requirements and implementation artifacts, since they are mostly concentrated at the bottom of the table. The fact that the term "requirements" is ranked first in the list is in accordance with the literature (Borg et al. 2013); the same goes for "source code" in the sense that it is the only software artifact that cannot be skipped in the software development lifecycle. The fact that classes are at 3rd place, denotes that most traceability studies consider object-oriented systems, whereas the explicit reference to use cases and UML diagrams suggest that there is also a preference in terms of design language and software development process.

Table 6.4: Most frequently traced software artifact types

Artifact Types	**Development Phase**	**Count**
Requirements (high /low level)	R	76
Source Code	I	60
Classes	I	42
UML Diagrams	D	33
Use Cases	R	31
Test Cases	T	29
Features/ Functional requirements	R	24
Methods/Functions/Operations	I	20
Design Models	D	10
Bug Reports / Issues	T	9
Informal specification / NL requirements	R	9
Code Elements (objects / attributes)	I	8
Design Artifacts	D	7
Architecture Documentation / Description / Decision /	D	7
Architectural Models (HL design)	D	6
Architectural Elements / Artifacts	D	6

Subsequently in Table 6.5, we present the top-5 most studied artifacts per development phase by re-grouping the results of Table 6.4 and including artifacts with lower counts. We observe that for each phase, the generic artifact is the dominant one (e.g., requirement, design, source code, test case).

Table 6.5: Most frequently traced software artifacts per development phase

Dev. Phase	Artifact	Count	Dev. Phase	Artifact	Count
R	Requirements (in general)	76	I	Source Code	55
	Use Cases	31		Classes	42
	Features / Functional requirements	20		Methods / Functions / Operations	20
	Informal Specifications / NL Requirements	9		Code Elements	8
	Concerns	5		Packages	3
D	UML Diagrams (in general)	12	T	Test cases	27
	Interaction Diagrams	12		Bug reports/ issues	9
	Design Models	10		Unit tests	4
	Design Artifacts	7		Tests	4
	Architectural Models (HL design)	6		Bugs	3
	Architectural Elements / Artifacts	6		Exceptions / Execution Errors	3

Next, we focus on pairs of artifacts, and we present in Table 6.6, the top-15 most frequently connected artifacts. Similarly to Table 6.4, we additionally present the phase of the artifact. One important observation from Table 6.6 is that the 6 most frequent pairs are between requirements, implementation, and testing artifacts. This observation is interesting for two reasons: (a) it makes sense as a sequence: first a

requirement is specified, then the code for that requirement is written (requirements-to-code traceability is the most frequent type of trace in the literature (Borg et al. 2013), and finally the code is tested, either starting from source code, or the tested requirement; (b) testing artifacts that were rather underrepresented in Table 6.4, rank very high in terms of pairs of artifacts. Another interesting observation is that a connection between artifacts produced in the same phase is rather uncommon. A more detailed view of the relations between pairs of artifacts is presented in Appendix A2, in a similar way to Table 6.6.

Table 6.6: Most Frequently Traced Pairs of Software Artifacts

Artifact 1	**Artifact 2**	**Development Phases**		**Count**
Requirements	Source Code	R	I	21
Use Cases	Classes	R	I	16
Requirements	Classes	R	I	14
Classes	Test Cases	I	T	10
Requirements	Test Cases	R	T	10
Source Code	Test Cases	I	T	7
Interaction Diagrams	Test Cases	D	T	6
Interaction Diagrams	Classes	D	I	6
Use Cases	Test Cases	R	T	6
Use Cases	Interaction Diagrams	R	D	6
High Level Requirements	Low Level Requirements	R	R	6
Source Code	Specifications	I	-	5
Features	Source Code	R	I	5
Requirements	Methods	R	I	5
Requirements	Design Models	R	D	5
Requirements	Requirements	R	R	5

6.3.2 Goals of Traceability Approaches and their Success Indicators (RQ_2)

This section summarizes the goals of the proposed approaches for traceability management. In Table 6.7, we classify the studies according to their goal, and also list their corresponding frequencies, as well as the most frequent success indicators. We note that some studies are classified under two categories: e.g., a study that introduces a novel approach can also compare it with existing ones. The analysis has led to four main categories: (a) studies that aim at proposing approaches for establishing traces, (b) studies that investigate the impact on traceability on quality attributes, (c) studies that investigate trace maintenance and evolution, (d) studies that propose benchmarks and guidelines:

- ***Establishment of Traces***: The majority of studies in Table 6.7 have the general goal of proposing novel traceability techniques or improving existing ones. In total 80 studies have proposed new techniques aiming to improve the accuracy of trace extraction, whereas 32 compared existing approaches to new ones or existing ones against each other. Finally, 16 papers proposed changes or the reconfiguration of existing approaches for improving their accuracy.

- ***Quality Attribute Important for Traceability:*** The second cluster of studies discusses the most important quality attributes while establishing traceability approaches. The most common qualities that are studied are the cost for establishing the traces, and the performance (usually time) for executing automated methods for trace recovery. We note that cost and performance are both measured in time but they differ in nature: cost is the time required to develop the traces manually, whereas performance refers to the time required by tools to automatically recover the traces. Also, usability of the detection approaches and the traces per se are highlighted by the community as of significant importance. Finally, we have identified two studies that deals with the scalability of automatic trace recovery.

- ***Trace Maintenance and Evolution:*** The third and much smaller set includes studies that deal with the evolution of traces along time and how they should be managed, as well as the visualization of traces.

- ***Other:*** the remaining set includes 6 studies that aimed at the development of benchmarks and guidelines to be used in traceability studies, as a means for validating approaches for establishing traces.

Table 6.7: Classification of general goals and focus of studies

General Goal	Focus on	Count	Success Indicators
Approaches Establishing Traceability Links	Novel Methods	80	Precision
	Comparison	32	Recall
	Improvement	16	F-measure
Quality Attribute important for Traceability	Cost	18	Time
	Performance	16	
	Usability	8	Positive Feedback from Users
			Time
	Scalability	2	# traces
			# Classes
Trace Maintenance	Evolution	9	#changes
	Representation / Visualization	7	User Answers
Other	Benchmarks or Guidelines	6	None

Next, in order to investigate if there are any chronological trends in the research focus of primary studies, we present in Figure 6.1 the chronological trends of the topics that have more than 10 studies. Regarding the proposal of new techniques, there was a large growth of papers after 2008, which later on stabilized. The rest of the groups of studies have not shown such a significant explosion in terms of number of studies, but are rather steadily increasing their numbers over the years.

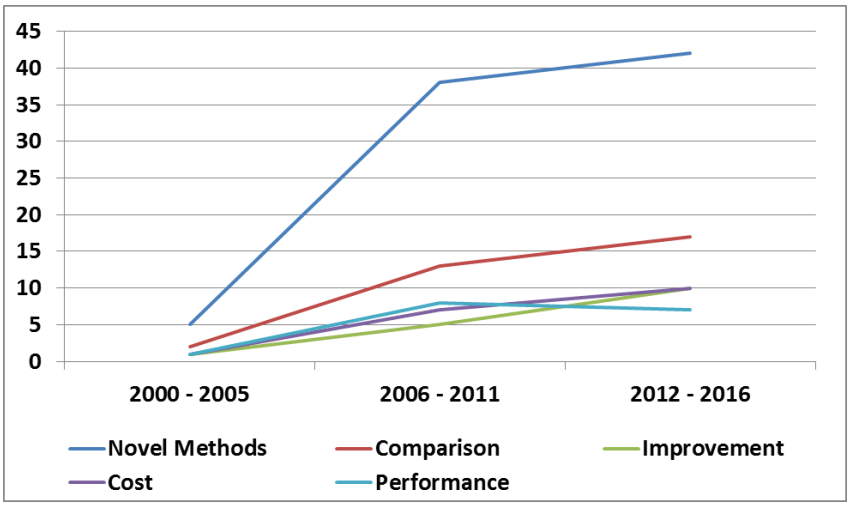

Figure 6.1: Chronological trends in the topics with count >10

6.3.3 Quality Attributes that Benefit from Traceability (RQ_3)

Table 6.8 lists a number of quality attributes (as defined in ISO25010) that are affected by traceability. Similarly as before, each quality is accompanied by the most common success indicators.

- ***Extendibility***: The relation between traceability and software extendibility (i.e., how easily a feature can be added into a system with or without traces) has been considered of great significance. For example, linking requirements to source code aids in identifying the classes (or code artifacts) that need to be updated upon a feature request.

- ***Correctness***: For the correctness of the software, traceability links are beneficial both in terms of debugging and writing defect-free code (14 studies). For example, the existence of traces can reduce the time required to spot the class (or code artifact) that contains the bug, and therefore facilitate the easy correction of errors.

- ***Understandability***: According to the primary studies, understandability can refer both to the ease of understanding a piece of software that has traces, and the ease of understanding the traces per se. For example, if a source code chunk (method or class) is linked to a requirement (which is usually more understandable, in

the sense that it is expressed in natural language), then it is more easy to understand, compared to non-linked artifacts.

- ***Testability***: This concerns whether software systems that have traces are easier to test (for instance, due to the existence of links between requirements and test cases). In particular, the primary studies suggest that linking artifacts to source code (ideally linking source code to test cases), the amount of source code that needs to be tested for a particular requirement is significantly reduced.

- ***Instability***: The literature suggests that whenever a trace is available, it is possible to perform more accurate change impact analysis, i.e., to identify which parts are affected when another part of the software is changed (i.e. instability). Therefore, the maintainability of the software is boosted.

Table 6.8: Impact of traceability approach on software, in terms of affected Qualities

Quality Characteristics	**Count**	**Success Indicators**
Extendibility	15	Time to add a feature
Correctness	14	#Errors Debugging time
Understandability	10	Positive Users' Feedback w.r.t. Understandability
Testability	5	LoC to be tested
Instability	4	Time to perform maintenance actions

Regarding the chronological trends on quality attributes, in Figure 6.2, we can observe that in all of the cases there is an increase along time. Therefore, we can assume that the influence of traces on quality is becoming a very relevant research and industry topic as the state of the art becomes more mature.

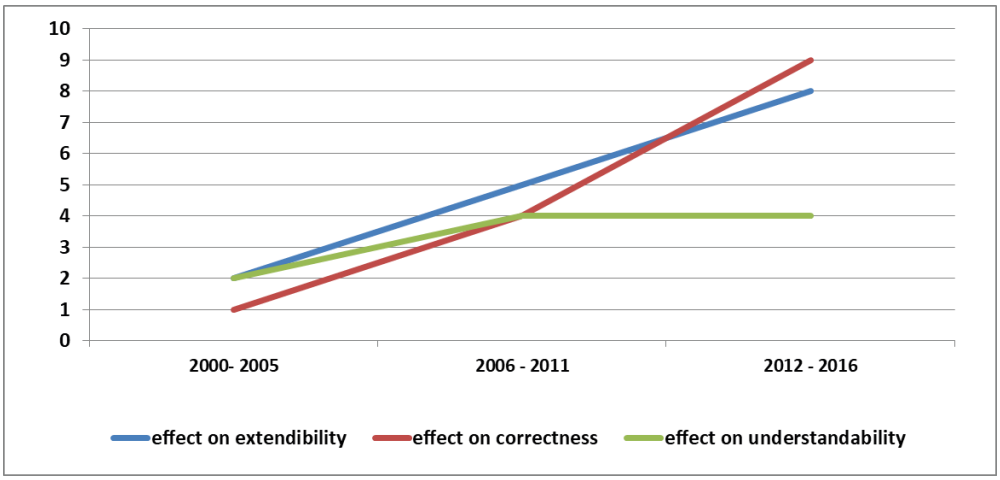

Figure 6.2. Chronological trends in the top 5 topics

6.3.4 Research Methods used (RQ_4)

In this Section we present our findings regarding the research methods that have been used for validating either the methods for establishing traces, or assessing how traces can influence quality. In Figure 6.3, we present some demographics of the research methods, whereas in Figure 6.4 the corresponding trends in time. The results of our study suggest that the majority of empirical studies are set up on observing the phenomenon in its current environment (case studies), followed by studies that control the environment in the form of an experiment. Regarding trend analysis, we can observe that proof-of-concept studies and simulations are constant over the years. Additionally, from 2009 onwards, surveys started to appear.

By cross-tabulating the pairs of development phases in which artifacts are traced and the employed research methods, we can observe that the majority of case studies include at least one artifact produced at the requirements engineering phase, whereas the majority of experiments include implementation artifacts. This finding can be explained by the fact that the requirements community typically performs more qualitative studies that rely on expert opinion and investigate the phenomenon in its real context. On the other hand, the source code community has a preferences for quantitative studies where typically source code repositories are mined to retrieve artifacts, then the artifacts are changed in a controlled way and then evaluated. A more detailed view on the exact artifacts being examined with different types of studies in presented in Appendix A2.

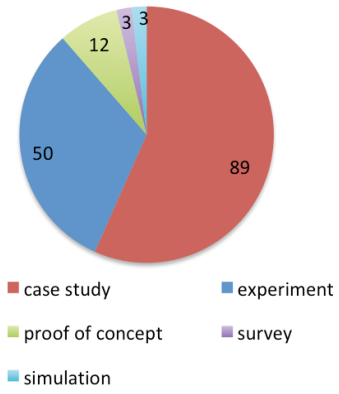

Figure 6.3. Count of research methods used

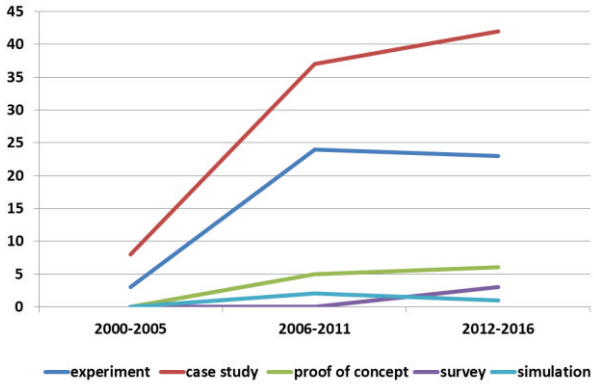

Figure 6.4. Chronological trends in the used research methods

Regarding the methods used in order to evaluate specific empirical goals, in Table 6.9, we present the cross tabulation of study goals with research methods. From the results of Table 6.9, we can observe that to evaluate the accuracy of novel trace recovery approaches, case studies are usually performed. This result is expected since real-world projects are used as units of analysis, and the actual traces are used as the golden standard; therefore, there is no need to control other factors. Additionally, to assess the cost of using traces, experiments is the most common research method.

Table 6.9: Cross tabs of study goals with research methods

Research Method	**Goal**	**Count**
Case Study	Novel Methods	115
	Improvement	45
	Comparison	43
	Cost	23
	Performance	23

Research Method	Goal	Count
Experiment	Cost	92
	Evolution	84
	Novel Methods	67
	Comparison	28
	Factor	20
Proof of Concept	Usability	15
	Benefits	13
Survey	Validation	4
	Benefits	1
Simulation	Novel Methods	2
	Performance	2

Based on Figure 6.5, we can observe that the majority of traceability studies employ quantitative analysis (72%), 16% facilitates qualitative analysis, whereas both qualitative and quantitative research methods is employed in 11% of the studies. This difference, suggests that there is a need for more qualitative studies that build upon experts' opinion through interviews, focus groups, and other data collection methods that rely on natural language instruments, and allow the participants of the study to express their view on the phenomenon under study. In most of the cases, the analysis has been performed with artifact analysis, and numerical data points.

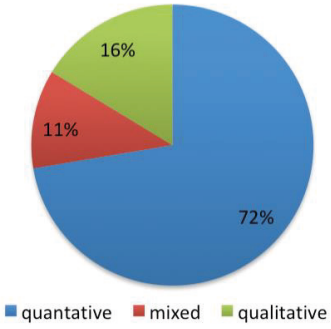

Figure 6.5. Frequencies on the types of data analysis methods

6.4 Discussion

In this section we recap the main findings of this study and discuss them, by comparing them to existing evidence coming from the literature, and by providing useful implications for researchers and practitioners.

Most Traced Artifacts. The most often traced artifacts are *software requirements*, followed by *source code*. The combination of the two artifacts is also the most commonly traced pair in the literature. One potential explanation is that these two artifacts are the most common ones used in practice: regardless of the development process used, requirements are formed in some kind (natural language, use cases, user stories, etc.) and of course source code is developed. Also, the majority of the studies that discuss or identify benefits of traceability use these two artifacts as examples (e.g., Maia and Lafeta 2013; Ali et al. 2015; Mäder & Egyed 2015 etc.). In addition, requirements and source code represent the problem and the solution respectively. During maintenance and evolution, both the problem and the solution are constantly updated, so traces among them facilitates changing one to reflect changes on the other.

Another interesting observation from the data is that for each phase, the generic artifact is the dominant one (e.g., requirements, design, source code, test cases) instead of specific artifacts (e.g. use cases, interaction diagrams, classes). This leads to the conclusion that current traceability approaches are not very fine-grained; for example they may aim at connecting source code to another artifact, without specifying the level of granularity (e.g., class, method, package, etc.). This more coarse-grained approach on the one hand, favors generality (i.e., the approach can be applied to all source code elements); on the other hand it might ignore characteristics of the specific artifacts. Additionally, although the approach is introduced at a coarse-grained artifact (e.g. source code level), the validation is performed only on one type of this artifact (e.g. classes), posing a threat to the validity of the obtained empirical evidence.

Goals of Traceability Approaches. Regarding the research goals of studies on traceability, our results suggest that the *introduction of novel traceability approaches* is the most common one in the literature. These approaches are empirically validated, usually in a quantitative manner, using precision and recall as success indicators. However, there are also many studies that only focus on *comparing existing approaches*, without proposing a novel one. Although the contribution of such studies at a first glance seems lower compared to proposing a novel approach,

we find that comparative studies have merit because: (a) as the corpus of research in traceability approaches grows, it is more relevant to compare them and identify scenarios in which every approach can be used, instead of adding new ones, to an already broad set of existing approaches; and (b) in empirical software engineering research, re-evaluation of methods and tools, by independent researchers (not the authors of the original studies) is very important for reliability, repeatability, and generalizability purposes. The work on proposing novel approaches, improving existing ones, or comparing approaches, highlights the paramount importance of having a baseline, through which approaches can be objectively contrasted in terms of identification accuracy. It is thus a healthy sign, that in the last years, we have identified a decent amount of studies that propose *benchmarks and guidelines for evaluating traceability approaches*.

The second large line of research in our results is the identification of quality attributes that are used to assess traceability approaches. On the one hand, automated approaches that aim at identifying traces after-the-fact are only considered useful in practice, if their performance is acceptable. On the other hand, approaches that involve humans in the detection or trace establishment can be very resource-intensive and this cost may prohibit their use in practice. Therefore, we advise both practitioners and researchers, to consider these two parameters (performance of automated approaches and cost of manual approaches) very carefully. As far as researchers are concerned, apart from the accuracy of the proposed techniques and the benefits of using traces, they should also consider the cost and/or performance of the proposed approach, to determine the cost-benefit ratio. Second, practitioners should place special emphasis on these two parameters in order to take informed decisions on selecting a traceability approach.

Finally, the results of the study suggest that trace development is not the only aspect that concerns researchers, but also what happens after traces have somehow been established. The research efforts are grouped in two categories. The first has to do with the visual representation of traces, focusing especially on their usability (e.g., a GUI-based information, inside the IDE) and striving for a smooth learning curve. The second, deals with the management of traces, dealing with when to update traces, where to store them, in which file format etc. This is a clear implication for researchers: they need to ensure that these two key requirements (usability and management of traces) drive the development of their tools, so as to increase the chances of industrial adoption. Additionally, practitioners can exploit this line of

research to identify the most user-friendly methods and tools, as well as those that provide the most fitting trace management features to their needs.

Benefit to Quality Attributes. We have explored the potential benefits that traceability brings to the software in terms of quality attributes. The main outcome of this investigation is that traceability, although it might increase development cost (due to the extra effort to establish the traces), it can provide significant benefits along maintenance. Therefore, most of the quality attributes that have been explored as beneficial to traceability are maintenance-related: *extendibility*, *correctness*, *instability*, *testability*, and *understandability*. The positive correlation of traceability and maintainability constitutes traceability as a very promising solution to one of the most important problems in the software engineering domain, i.e., the increasing maintenance costs. According to van Vliet (2008), maintenance cost can sum up to 50% of the total cost of software development, and can increase by a further 75% (Galorath 2008) when the software is instable and produces ripple effects, i.e., changes in one part of the system causes changes in other parts (Arvanitou et al. 2015). The trade-off between the costs occurring while developing traces and the financial benefits obtained along maintenance can be quantified through cost-benefit analysis. This can be an important factor in applying traceability, as the return of investment during maintenance can be a decisive factor in convincing project managers to spend resources on traceability.

Empiricism in Traceability Research. The level of empirical evidence in traceability research is quite high, in the sense that many case studies and experiments have been already conducted (levels 4 to 6—out of 6, according to Alves et al. (2010)). Also, a number of proof-of-concept applications have been demonstrated, so as to showcase the benefits of traceability in practice. Finally, surveys have recently started to appear. This later start of surveys compared to the other methods may be because the motivation to use surveys as research methods relies on the existence of knowledge or experience of the subjects with the topic. Indeed, the application of traceability approaches in the industry was until recently rather limited (Spanoudakis and Zisman 2005); however the execution of surveys in the past years may be an indication of a more wide-spread application of such approaches in practice.

The overall level of evidence is encouraging, especially as it combines quantitative results from experiments (including several causal investigations) with qualitative results from case studies (including several explorative investigations). Research methods that are underused for the time being, are longitudinal industrial case stud-

ies and action research. Such studies, if rigorously conducted, would significantly increase the industrial relevance of traceability research and contribute towards their further adoption in practice.

6.5 Related Work

In this section we present related work in terms of other secondary studies in the domain of software artifact traceability, and subsequently summarize and compare this work with our study. In 2005, Spanoudakis and Zisman (2005) presented a software traceability roadmap which reviewed and presented (a) different classifications of traceability relations, (b) different approaches for generating, representing, recording, and maintaining traceability relations, and (c) different ways of deploying traceability relations in the software development process. In 2007, Galvao and Goknil (2007) conducted a review of the state-of-the-art on the topic of traceability approaches in MDE. The authors analyzed the primary studies with respect to five general comparison criteria: representation, mapping (i.e. traceability of model elements at different levels of abstraction), scalability, change impact analysis and tool support. In 2010, Winkler and Pilgrim (2010) investigated traceability similarities and differences also in the domains of model driven and software engineering. Their research provided a basic description of traceability and associated topics, elaborating on how traceability can be achieved and used. Also, they investigated the state-of-practice and the limits of traces application in the industry.

Torkar et al. (2012) conducted a systematic literature review on requirements traceability. The study considers primary studies during the period 1997 to 2007 and aims at answering two main research questions, regarding: (a) the existing definitions of the requirements' traceability, and (b) the existing requirements' traceability techniques, their challenges and the related tools found in literature. In 2017, Tufail et al. (2017) performed a Systematic Literature Review also in the area of Requirements Traceability aiming to identify the leading models, challenges and tools in the domain during 2010 to 2017, as well as the pros and cons of the leading requirement traceability models and tools. In the same year, Omar and Dahr conducted also a systematic literature review on the same field, aiming to present to practitioners interested into finding a suitable method for tracing requirements, the most recent (from 2008-2017) requirement traceability practices and tools available (Omar and Dahr 2017). Regan et al. (2012a) conducted a review on traceability analyzing the motivations of the organizations for implementing traceability mostly within the regulated software safety critical domain, but also

by firms outside of this field. The same year they published a second review on the topic of traceability from the perspective of the implementation of traceability in real organizations. This study aimed at presenting the barriers faced by organizations while implementing traceability. Additionally it presented proposed solutions to these barriers, while it provided a comparison of them, for organizations operating inside and outside of the domain of critical systems. (Regan et al. 2012b)

Another Systematic Mapping Study was conducted by Borg et al. focused on information retrieval (IR)-based trace recovery approaches. The scope of the study is limited, focusing only on traceability approaches of natural language (NL) software artifacts, during the years 1999 to 2011. The research questions that are investigated during the study consider (a) the identification of the most frequent IR approaches for tracing NL software artifacts, (b) the types of the artifacts that are most commonly linked, and (c) the level of evidence during the evaluation of these approaches (Borg et al. 2013). In 2013 one more review on the topic of traceability were published. Nair et al. (2013) collected 70 primary studies on the domain of traceability, which had been published in the requirements engineering conference in a period of 20 years. The review investigated what traceability related aspects have been studied and by whom, what types of systems have been considered, what types of artefacts have been traced and what empirical methods have been applied. Additionally, the study reported on specific challenges on the domain that have been addressed by primary studies, and tool features that have been developed to support traceability. In 2014, Javed and Zdun conducted an SLR aiming to discover the existing traceability approaches and tools between software architecture and source code, as well as the empirical evidence for these approaches, their benefits and liabilities, their relations to software architecture understanding, and issues, barriers, and challenges of the approaches (Javed and Zdun 2014). Finally, Cleland-Huang et al. (2012) conducted a roadmap for software and system traceability research, through which they present a brief view of the state of the art in traceability, the biggest challenges identified and future directions for the field. However, although this work reports on existing studies on Traceability Information Models, Automated Trace Creation and Maintenance and Traceability Economics, and provides directions based on the existing literature in the field of traceability, it is not directly comparable to our work, because it stays on a higher level.

There are several points of differentiation between the aforementioned studies and our work. An overview of the comparison between these studies and our work is presented in Table 6.10. In particular, for every secondary study, we present its

scope (i.e., what domain or paradigm it focuses on, what types of artifacts it investigates, which venues were searched and until which year primary studies were collected) and whether it covers the research questions of our own study (i.e., whether it studies the type of artifacts most commonly linked, the goals of traceability approaches, the benefits and the research methods used). As indicated by the table, our study has performed an analysis on the biggest dataset of primary studies compared to previous related work. Also, although our study focused only on empirical studies, there were no restrictions applied in terms of a specific domain or concrete artifacts under investigation.

6.6 Threats to Validity

In this section we present threats to validity and their mitigation, based on the guidelines provided by Ampatzoglou et al. (2019). Specifically, in Section 6.6.1, we report threats to validity related to study selection, in Section 6.6.2, we report threats related to data validity, and in Section 6.6.3, we report threats related to research validity.

6.6.1 Study Selection Validity

Study selection validity concerns the early phases of the research, i.e. the search process and the filtering of studies. In order to ensure that our search process has adequately identified all relevant studies, the primary studies that have been selected for inclusion have been carefully chosen following a well-defined protocol based on strict guidelines (Kitchenham and Charters 2007). The identification procedure consisted of an automated search through the search engines of the most-known DLs. The search string that we used (see Section 6.2.2) is quite broad, since we only included the name of the investigated research method and synonyms of traceability, aiming to retrieve the maximum number of relevant studies. However, studies that adopted different terminology than the most established ones might have been excluded. The benefit of focusing only on research efforts that use standard terminology, is that we avoided using subjective criteria for characterizing the type of empirical research.

Next, during the article inclusion/exclusion phase, there is always a possibility of excluding relevant articles. To mitigate this threat, two researchers have been involved in this process, discussing any possible conflicts. On the completion of this

Table 6.10: Comparison with related work

Secondary Study	Scope					Goals			
	Domain / Paradigm	**Connected Artifacts**	**Search Venues**	**End Year**	**#Primary Studies**	**Traced Artifacts**	**Goal of Studies**	**Quality Attributes**	**Research Method**
Spanoudakis and Zisman (2005)		All to all	Not mentioned	Not mentioned	Not mentioned	YES	NO	NO	NO
Galvao and Goknil (2007)	MDE	All to all	Not mentioned	Not mentioned	Not mentioned	NO	NO	NO	NO
Winkler and Pilgrim (2010)	MDE + Require-ments		main confer-ences, work-shops, and jour-nals of both the RE and MDD communities + snowballing	Not mentioned	Not mentioned	NO	NO	NO	NO
Torkar et al. (2012)		Require-ments to other arti-facts	5 databases	2007	29	NO	NO	NO	NO

Tufail et al (2017)	Require-ments	Require-ments	5 databases	2017	33	YES	NO	NO	NO
Omar and Dahr (2017)	Require-ments		5 databases	2015	37	NO	YES	NO	NO
Regan et al. (2012a)	(Critical Systems)		Not mentioned	Not mentioned	8	NO	NO	NO	NO
Regan et al. (2012b)	(Critical Systems) – Case Stud-ies only		2 database + Google scholar	Not mentioned	8	NO	NO	NO	NO
Borg et al. (2013)	IR tracea-bility approaches	NL artifacts to other artifacts	6 databases	2011	79	YES	NO	NO	YES
Nair et al. (2013)		All to all	The Require-ments Engineer-ing Conference	2012	70	YES	YES	NO	YES
Javed and Zdun (2014)		Architec-ture and source code	1 database + snowballing	2013	11	NO	YES	NO	YES
Our SMS	Empirical studies	All to all	4 databases	2016	156	YES	YES	YES	YES

process, a third researcher was randomly screening the selection of articles for inclusion. Also, the inclusion/exclusion criteria have been extensively discussed among the authors, so as to guarantee their clarity and prevent misinterpretations. Furthermore, from our searching space we have excluded grey literature, since the goal of the study focuses on the use of empirical evidence, which are almost never published in grey literature.

Additionally, although we have not identified any duplicate articles, our research protocol dictated that we check for duplicated articles, based on the abstract. Upon identification of duplicates, the most extensive one would be retained. Also, our study is not suffering from missing non-English papers and the papers published in a limited number of journals and conferences, since our search process was aiming at a large number of publication venues (including DLs as a whole) all publishing papers only in English. Finally, we have been able to access all publications that we were interested in, since our research institutes provide us access to the used DLs.

6.6.2 Data Validity

Regarding data validity, the main threat is related to data extraction bias. All relevant data were extracted and recorded manually by the third author. Since this procedure is prone to some subjectivity, two researchers further inspected and refined the collected data, re-validating them. After this procedure the results were discussed among all researchers and any conflicts were resolved. Additionally, no publication bias is present in our results since primary studies have been collected from various venues. Thus, we believe that the obtained data points are not influenced by a small group of people.

Our secondary study is not affected by the following threats: (a) small sample size—we have been able to retrieve approx. 100 articles; (b) lack of relationships—our study was not aiming to identify any relationships among data, but only to classify and synthesize; (c) low quality of primary studies—since quality assessment is not relevant for SMSs; and (d) selection of variables to be extracted—the straightforward research questions of our study have not raised any conflicts in the discussions among authors on which variables should be extracted. Moreover, we did not identify issues with the use of statistical analysis, in the sense that the nature of our research questions did not require hypothesis testing but only basic statistical analysis (descriptive statistics). Finally, to mitigate the researchers' bias in data inter-

pretation and analysis the authors have discussed the data clustering for the goal of the studies, the qualities of interest, and the research methods used.

6.6.3 Research Validity

Concerning research validity, the relevant threats concern research method bias and repeatability. Regarding the former, the authors are highly familiar with the process of conducting secondary studies, since they have been involved in a large number as authors and reviewers. Therefore, no mitigation actions were necessary. Regarding the latter, we believe that the followed review process ensures the reliability and the safe replication of our study. First, all important decisions in our review planning have been thoroughly documented in this manuscript (see Section 6.2) are can be easily reproduced by other researchers. Second, the fact that the data extraction was based on the opinion of three researchers can to some extent guarantee the elimination of bias, making the dataset reliable. Third, all extracted data have been made publicly available, so as to enable comparison of results.

Additionally, through discussion among the authors we have set four research questions that accurately and holistically map to the set goal. This is clearly depicted by the mapping of each research question to the research sub-goals/objectives. Furthermore, in the literature we have been able to identify a substantial amount of related works that can be used for comparison to our results. In particular, for this reason we used related studies from the software engineering literature. Finally, the selection of the research method is adequate for the goal of this study and no deviations from the guidelines have been performed.

6.7 Conclusions

This study focuses on software artifacts' traceability, i.e., the connection of artifacts. In particular, we aim at identifying studies that provide any kind of empirical evidence related to traceability, and understanding their characteristics (in terms of linked artifacts and research methods) and goals. To achieve this goal we have performed a systematic mapping study, which has led to the inclusion of 155 studies. The results of the study suggested that requirements and source code are the mostly studied software artifacts, a fact that can be explained due to their nature, and importance in the software development lifecycles. Regarding the goals of the studies, our results suggest that most of the studies aim at proposing novel traceability methods, whereas the most studied quality attributes that are affected by

traceability are maintainability-related. The outcomes are discussed in this chapter from various perspectives, and have resulted in useful implications for researchers and practitioners.

This chapter was the first step into exploring the problem space around Documentation TD, by reporting on the state-of-research on traceability techniques that may act as mechanisms for preventing requirements documentation TD. Based on the results of the study, and the lack of an existing approach for effectively preventing requirements traceability TD, in the next chapter we will proceed with proposing a requirements-to-code traceability technique that could be smoothly integrated in the daily routine of software engineers (e.g., by integrating it in the programming IDE) for this purpose. We will also validate the usefulness of the proposed technique by evaluating its influence on requirements documentation TD.